

# Distributed Markov Chains

Sumit Kumar Jha<sup>1</sup>, Madhavan Mukund<sup>2</sup>, Ratul Saha<sup>3</sup>, and P. S. Thiagarajan<sup>3</sup>

<sup>1</sup> University of Central Florida, USA, [jha@eecs.ucf.edu](mailto:jha@eecs.ucf.edu)

<sup>2</sup> Chennai Mathematical Institute, India, [madhavan@cmi.ac.in](mailto:madhavan@cmi.ac.in)

<sup>3</sup> National University of Singapore, Singapore, [{ratul,thiagu}@comp.nus.edu.sg](mailto:{ratul,thiagu}@comp.nus.edu.sg)

**Abstract.** The formal verification of large probabilistic models is important and challenging. Exploiting the concurrency that is often present is one way to address this problem. Here we study a restricted class of asynchronous distributed probabilistic systems in which the synchronizations determine the probability distribution for the next moves of the participating agents. The key restriction we impose is that the synchronizations are *deterministic*, in the sense that any two simultaneously enabled synchronizations must involve disjoint sets of agents. As a result, this network of agents can be viewed as a succinct and distributed presentation of a large global Markov chain. A rich class of Markov chains can be represented this way.

We define an interleaved semantics for our model in terms of the local synchronization actions. The network structure induces an independence relation on these actions, which, in turn, induces an equivalence relation over the interleaved runs in the usual way. We construct a natural probability measure over these equivalence classes of runs by exploiting Mazurkiewicz trace theory and the probability measure space of the associated global Markov chain.

It turns out that verification of our model, called DMCs (distributed Markov chains), can often be efficiently carried out by exploiting the partial order nature of the interleaved semantics. To demonstrate this, we develop a statistical model checking (SMC) procedure and use it to verify two large distributed probabilistic networks.

## 1 Introduction

We present here a class of distributed probabilistic systems called distributed Markov chains (DMCs). A DMC is a network of probabilistic transition systems that synchronize on common actions. The synchronizations are deterministic in the sense that two simultaneously enabled synchronization actions must involve disjoint sets of agents. The information that the agents gain through a synchronization determines the probability distribution for their next moves. Internal actions correspond to synchronizations involving only one agent.

In many distributed probabilistic systems, the communication protocol can be designed to be deterministic—some examples are discussed in Section 8. Hence, the determinacy restriction is less limiting than may appear at first sight.

We define an interleaved semantics where one synchronization action is executed at a time. The resulting object is, in general, *not* a Markov chain. Thus,

defining a valid probability measure over interleaved runs—called *trajectories*—is a technical challenge. We address this by noting that there is a natural independence relation on local actions—two actions are independent if they involve disjoint sets of agents. Using this relation, we partition the trajectories into equivalence classes. As usual, each equivalence class will correspond to a partially ordered execution. This leads to a trajectory space that resembles the usual path space of a Markov chain [3], except that it will not be tree-like. Hence, one cannot readily define a probability measure over this space.

Due to the determinacy restriction, at any global state any two enabled actions will be independent. Thus, by letting all the enabled actions at a global state occur as a single step followed by probabilistic moves by all the involved agents, one obtains a Markov chain.

Using Mazurkiewicz trace theory [8], we then embed the trajectory space derived from the interleaved semantics into the path space of the global Markov chain. This induces a probability measure over the trajectory space. This is the key technical contribution of the paper. We are not aware of a similar result for any well-defined class of distributed probabilistic systems [1, 2, 22, 23].

Due to its exponential size (in the number of agents), it will often be infeasible to analyze a DMC in terms of its global Markov chain. In contrast, due to the partial order nature of the trajectory space, the global behaviour of the network can often be efficiently analyzed using the interleaved semantics. To bring this out, we formulate a statistical model checking (SMC) problem for DMCs in which the specifications consist of boolean combinations of local bounded linear temporal logic (BLTL) [3] formulas. We then develop a sequential probability ratio test (SPRT) based SMC procedure [25, 26] to solve this problem. We view our SMC procedure as a first step. Other partial order based reduction techniques such as ample sets [6] and finite prefixes of unfoldings [9] can also be readily developed for DMCs. Furthermore, one can develop model checking procedures for more powerful specification logics.

We illustrate the potential of our approach by using our SMC procedure to analyze two distributed probabilistic algorithms. The first is a distributed leader election protocol in an anonymous ring [14]. The second one is a randomized solution to the classical dining philosophers problem [20]. We show that for DMCs, simulations based on asynchronous trajectories are significantly faster than working directly with the global state space.

To summarize, our main contribution is identifying determinacy of communications as a fruitful restriction for distributed stochastic systems and constructing a probability measure over a partially ordered space of runs. We believe DMCs represent a restricted but clean combination of concurrent and stochastic dynamics and can lead to fruitful applications in domains such as embedded systems [12, 15, 19], biological processes [5, 18], and distributed protocols [7, 16].

**Related work** Our work is in line with partial order based methods for Markov Decision Processes (MDPs) [11] where, typically, a partial commutation structure is imposed on the actions of a *global* MDP. For instance, in [4], partial order reduction is used to identify “spurious” nondeterminism arising out of the

interleaving of concurrent actions, in order to determine when the underlying behaviour corresponds to a Markov chain. In contrast, in a DMC, deterministic communication ensures that local behaviours *always* generate a global Markov chain. The independence of actions is directly given by the local state spaces of the components. This also makes it easier to model how components influence each other through communications.

The interplay between concurrency and stochasticity has also been explored in the setting of event structures [1,23]. In these approaches, the global behaviour—which is *not* a Markov chain—is endowed with a probability measure. Further, probabilistic verification problems are not formulated and studied. Markov nets, studied in [2] can be easily modeled as DMCs. In [2], the focus is on working out a probabilistic event structure semantics rather than developing a model checking procedure based on the interleaved semantics, as we do here.

Our model is formulated as a sub-class of probabilistic asynchronous automata [22], where we require synchronizations to be deterministic. This restriction allows us to develop a probability measure over the (infinite) trajectory space, which in turn paves the way for carrying out formal verification based on probabilistic temporal logic specifications. In contrast, the work reported in [22] is language-theoretic, with the goal of generalizing Zielonka’s theorem [28] to a probabilistic setting. However, in the model of [22], conflicting actions may be enabled at a global state and it is difficult to see how one can formulate a  $\sigma$ -algebra over the runs with a well-defined probability measure.

## 2 The Distributed Markov Chain (DMC) Model

We fix  $n$  agents  $\{1, 2, \dots, n\}$  and set  $[n] = \{1, 2, \dots, n\}$ . For convenience, we denote various  $[n]$ -indexed sets of the form  $\{X_i\}_{i \in [n]}$  as just  $\{X_i\}$ . We begin with some notation for distributed state spaces.

**Definition 1.** For  $i \in [n]$ , let  $S_i$  be a finite set of local states, where  $\{S_i\}$  is pairwise disjoint.

- We call  $S = \bigcup_i S_i$  the set of local states.
- For nonempty  $u \subseteq [n]$ ,  $\mathbf{S}_u = \prod_{i \in u} S_i$  is the set of  $u$ -states.
- $\mathbf{S}_{[n]}$  is the set of global states, which we typically denote  $\mathbf{S}$ .
- For a state  $\mathbf{v} \in \mathbf{S}_u$  and  $w \subseteq u$ ,  $\mathbf{v}_w$  denotes the projection of  $\mathbf{v}$  to  $\mathbf{S}_w$ .
- For  $u = \{i\}$ , we write  $\mathbf{S}_i$  and  $\mathbf{v}_i$  rather than  $\mathbf{S}_{\{i\}}$  and  $\mathbf{v}_{\{i\}}$ , respectively.

Our model is a restricted version of probabilistic asynchronous automata [22].

**Definition 2.** A probabilistic asynchronous system is a structure  $(\{S_i\}, \{s_i^{in}\}, A, loc, en, \{\pi^a\}_{a \in A})$  where:

- $S_i$  is a finite set of local states for each  $i$  and  $\{S_i\}$  is pairwise disjoint.
- $s_i^{in} \in S_i$  is the initial state of agent  $i$ .
- $A$  is a set of synchronization actions.
- $loc : A \rightarrow 2^{[n]} \setminus \emptyset$  specifies the agents that participate in each action  $a$ .

$\mathbf{s}'$

- For  $a \in A$ , we write  $\mathbf{S}_a$  instead of  $\mathbf{S}_{loc(a)}$  and call it the set of  $a$ -states.
- For each  $a \in A$ ,  $en_a \subseteq \mathbf{S}_a$  is the subset of  $a$ -states where  $a$  is enabled.
- With each  $a \in A$ , we associate a probabilistic transition function  $\pi^a : en_a \rightarrow (\mathbf{S}_a \rightarrow [0, 1])$  such that, for every  $\mathbf{v} \in en_a$ ,  $\sum_{\mathbf{u} \in \mathbf{S}_a} \pi^a(\mathbf{v})(\mathbf{u}) = 1$ .

The action  $a$  represents a synchronized communication between the agents in  $loc(a)$  and it is enabled at the global state  $\mathbf{s}$  if  $\mathbf{s}_a \in en_a$ . When  $a$  occurs at  $\mathbf{s}$ , only the components in  $loc(a)$  are involved in the move to the new global state  $\mathbf{s}'$ ; the new  $a$ -state  $\mathbf{s}'_a$  is chosen probabilistically according to the distribution assigned to the current  $a$ -state  $\mathbf{s}_a$  by  $\pi^a$ . For every  $j \notin loc(a)$ ,  $\mathbf{s}_j = \mathbf{s}'_j$ .

We would like to lift the probabilities associated with individual moves to a probability measure over runs of the system. This is difficult to achieve, in general, because of the combination of nondeterminism, concurrency and probability in the model. This motivates us to restrict the nondeterminism in the model.

For an agent  $i$  and a local state  $s \in S_i$ , we define the set of actions compatible with  $s$  to be  $act(s) = \{a \mid i \in loc(a), s = \mathbf{v}_i \text{ for some } \mathbf{v} \in en_a\}$ .

**Definition 3.** A distributed Markov chain (DMC) is a probabilistic asynchronous system  $\mathcal{D} = (\{S_i\}, \{s_i^{in}\}, A, loc, en, \{\pi^a\}_{a \in A})$  in which, for each agent  $i$  and each local state  $s \in S_i$ ,  $|act(s)| = 1$ .

In other words, in a DMC, the set of partners that an agent can communicate with next is fixed deterministically by its current local state. Hence, if two actions  $a$  and  $b$  are enabled at a global state  $\mathbf{s}$ , they must involve disjoint sets of agents—that is,  $loc(a) \cap loc(b) = \emptyset$ . This allows us to capture the global behaviour of the model as a Markov chain—as shown in Section 4—whence the name DMC.

**Events** Let  $\mathcal{D}$  be a DMC. An *event* of  $\mathcal{D}$  is a triple  $e = (\mathbf{v}, a, \mathbf{v}')$  where  $\mathbf{v}, \mathbf{v}' \in \mathbf{S}_a$ ,  $\mathbf{v} \in en_a$  and  $\pi^a(\mathbf{v})(\mathbf{v}') > 0$ . We extend  $loc$  to events via  $loc((\mathbf{v}, a, \mathbf{v}')) = loc(a)$ .

Suppose  $e = (\mathbf{v}, a, \mathbf{v}')$  is an event and  $p = \pi^a(\mathbf{v})(\mathbf{v}')$ . Then  $e$  represents an occurrence of the synchronization action  $a$  followed by a joint move by the agents in  $loc(a)$  from  $\mathbf{v}$  to  $\mathbf{v}'$  with probability  $p$ . Again, components outside  $loc(e)$  are unaffected by this move.

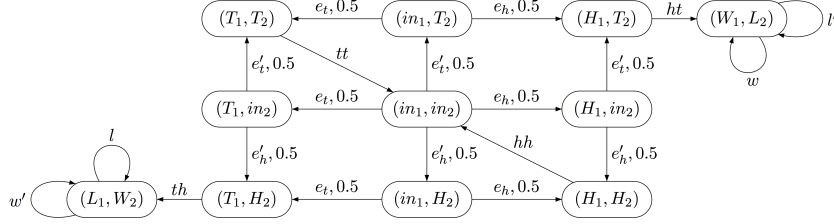
Let  $\Sigma$  denote the set of events of  $\mathcal{D}$  and  $e, e', \dots$  range over  $\Sigma$ . With the event  $e = (\mathbf{v}, a, \mathbf{v}')$  we associate the probability  $p_e = \pi^a(\mathbf{v})(\mathbf{v}')$ .

**The interleaved semantics** We now associate a global transition system with  $\mathcal{D}$  based on event occurrences.

Recall that  $\mathbf{S}$  is the set of global states. The event  $e = (\mathbf{v}, a, \mathbf{v}')$  is *enabled* at  $\mathbf{s} \in \mathbf{S}$  iff  $\mathbf{v} = \mathbf{s}_a \in en_a$ . The transition system of  $\mathcal{D}$  is  $TS = (\mathbf{S}, \Sigma, \rightarrow, \mathbf{s}^{in})$ , where  $\rightarrow \subseteq \mathbf{S} \times (\Sigma \times (0, 1]) \times \mathbf{S}$  is given by  $\mathbf{s} \xrightarrow{e, p_e} \mathbf{s}'$  iff  $e = (\mathbf{v}, a, \mathbf{v}')$  is enabled at  $\mathbf{s}$ ,  $\mathbf{s}'_a = \mathbf{v}'$  and  $\mathbf{s}_j = \mathbf{s}'_j$  for every  $j \notin loc(e)$ .

In Fig. 1 we show the transition system of a DMC describing a simple two player game. Each player tosses an unbiased coin. If the tosses have the same

$\mathbf{s}'$



**Fig. 1.** The transition system of a DMDP for a two player game

outcome the players toss again. If the outcomes are different then the player who tossed heads wins. In this 2-component system,  $S_i = \{in_i, T_i, H_i, L_i, W_i\}$  for  $i = 1, 2$ , where  $T_i/H_i$  denote that a tail/head was tossed, respectively, and  $L_i/W_i$  denote local losing/winning states, respectively. Agent 1, for instance, has an internal action  $a_1$  with  $loc(a_1) = \{1\}$ ,  $en_{a_1} = \{in_1\}$  and  $\pi^{a_1}(in_1)(T_1) = 0.5 = \pi^{a_1}(in_1)(H_1)$ . Thus  $e_h = (\{in_1\}, a_1, \{H_1\})$  and  $e_t = (\{in_1\}, a_1, \{T_1\})$  are both events that are enabled at  $(in_1, in_2)$ . Symmetrically, agent 2 has the internal action  $\{a_2\}$  with  $loc(a_2) = \{2\}$ ,  $en_{a_2} = \{in_2\}$  and corresponding events  $e'_h$  and  $e'_t$ . On the other hand,  $b$  is an action with  $loc(b) = \{1, 2\}$ ,  $en_b = \{(T_1, T_2)\}$  where  $tt = (\{T_1, T_2\}, b, \{in_1, in_2\})$  is an event with  $\pi^b((T_1, T_2))(in_1, in_2) = 1$ . To aid readability, if the probability of an event is 1 then this value is not shown.

**The trace alphabet**  $(\Sigma, I)$  We conclude this section by defining the independence relation  $I \subseteq \Sigma \times \Sigma$  given by  $e I e'$  iff  $loc(e) \cap loc(e') = \emptyset$ . Clearly  $I$  is irreflexive and symmetric and hence  $(\Sigma, I)$  is a Mazurkiewicz trace alphabet [8].

### 3 The trajectory space

Let  $TS$  be the transition system associated with a DMC  $\mathcal{D}$ . To reason about the probabilistic behaviour of  $TS$ , one must follow the technique used for Markov chains—namely, build a  $\sigma$ -algebra over the paths of this transition system, endowed with a probability measure. The major difficulty is that, due to the mix of concurrency and stochasticity,  $TS$  is not a Markov chain in general. In Fig. 1, for instance, the sum of the probabilities of the transitions originating from the state  $(in_1, in_2)$  is 2. To get around this, we will filter out concurrency by working with equivalence classes of paths rather than individual paths.

We refer to paths in  $TS$  as *trajectories*. A finite *trajectory* of  $TS$  from  $\mathbf{s} \in \mathbf{S}$  is a sequence of the form  $\mathbf{s}_0 e_0 \mathbf{s}_1 \dots \mathbf{s}_{k-1} e_{k-1} \mathbf{s}_k$  such that  $\mathbf{s}_0 = \mathbf{s}$  and, for  $0 \leq \ell < k$ ,  $\mathbf{s}_\ell \xrightarrow{e_\ell, p_\ell} \mathbf{s}_{\ell+1}$  (with  $p_\ell = p_{e_\ell}$ ). Infinite trajectories are defined as usual.

For the trajectory  $\rho = \mathbf{s}_0 e_0 \mathbf{s}_1 \dots \mathbf{s}_{k-1} e_{k-1} \mathbf{s}_k$ , we define  $ev(\rho)$  to be the event sequence  $e_0 e_1 \dots e_{k-1}$ . Again, this notation is extended to infinite trajectories in the natural way. Due to concurrency, one can have infinite trajectories that are not maximal, so we proceed as follows.

Let  $\Sigma_i = \{e \mid i \in loc(e)\}$ . Suppose  $\xi$  is an event sequence (finite or infinite). Then  $proj_i(\xi)$  is the sequence obtained by erasing from  $\xi$  all events that are not in  $\Sigma_i$ . This leads to the equivalence relation  $\approx$  over event sequences given by

$\mathbf{s}'$

$\xi \approx \xi'$  iff  $proj_i(\xi) = proj_i(\xi')$  for every  $i$ . We let  $[\xi]$  denote the  $\approx$ -equivalence class containing  $\xi$  and call it a (*Mazurkiewicz*) *trace*.<sup>4</sup> The partial order relation  $\sqsubseteq$  over traces is defined as  $[\xi] \sqsubseteq [\xi']$  iff  $proj_i(\xi)$  is a prefix of  $proj_i(\xi')$  for every  $i$ . Finally the trace  $[\xi]$  is said to be maximal iff for every  $\xi'$ ,  $[\xi] \sqsubseteq [\xi']$  implies  $[\xi] = [\xi']$ . The trajectory  $\rho$  is *maximal* iff  $[ev(\rho)]$  is a maximal trace. In the transition system of Fig. 1,  $(in_1, in_2)e_h(H_1, in_2)e'_T(H_1, T_2)ht((W_1, L_2)l')^\omega$  is a non-maximal infinite trajectory.

**The  $\sigma$ -algebra of trajectories** We denote by  $Trj_{\mathbf{s}}$  the set of maximal trajectories from  $\mathbf{s}$ . Two trajectories can correspond to interleavings of the same partially ordered execution of events. Hence one must work with equivalence classes of maximal trajectories to construct a probability measure. The equivalence relation  $\simeq$  over  $Trj_{\mathbf{s}}$  that we need is defined as  $\rho \simeq \rho'$  iff  $ev(\rho) \approx ev(\rho')$ . As usual  $[\rho]$  will denote the equivalence class containing the trajectory  $\rho$ .

Let  $\rho$  be finite trajectory from  $\mathbf{s}$ . Then  $\uparrow\rho$  is the subset of  $Trj_{\mathbf{s}}$  satisfying  $\rho' \in \uparrow\rho$  iff  $\rho$  is a prefix of  $\rho'$ . We now define  $BC(\rho)$ , the basic *trj*-cylinder at  $\mathbf{s}$  generated by  $\rho$ , to be the least subset of  $Trj_{\mathbf{s}}$  that contains  $\uparrow\rho$  and satisfies the closure property that if  $\rho' \in BC(\rho)$  and  $\rho' \simeq \rho''$  then  $\rho'' \in BC(\rho)$ . In other words,  $BC(\rho) = \{[\rho'] \mid \rho' \in Trj_{\mathbf{s}}, [ev(\rho)] \sqsubseteq [ev(\rho')]\}$ .

It is worth noting that we could have  $BC(\rho) \cap BC(\rho') \neq \emptyset$  without having  $\rho \simeq \rho'$ . For instance, in Fig. 1, let  $\rho = (in_1, in_2)e_h(H_1, in_2)$  and  $\rho' = (in_1, in_2)e'_t(in_1, T_2)$ . Then  $BC(\rho)$  and  $BC(\rho')$  will have common maximal trajectories of the form  $(in_1, in_2)e_h(H_1, in_2)e'_t(H_1, T_2) \dots$

We now define  $\widehat{SA}(\mathbf{s})$  to be the least  $\sigma$ -algebra that contains the basic *trj*-cylinders at  $\mathbf{s}$  and is closed under countable unions and complementation (relative to  $Trj_{\mathbf{s}}$ ).

To construct the probability measure  $\widehat{P} : \widehat{SA}(\mathbf{s}) \rightarrow [0, 1]$  we are after, a natural idea would be to assign a probability to each basic *trj*-cylinder as follows. Let  $BC(\rho)$  be a basic *trj*-cylinder with  $\rho = \mathbf{s}_0 e_0 \mathbf{s}_1 \dots \mathbf{s}_{k-1} e_{k-1} \mathbf{s}_k$ . Then  $\widehat{P}(BC(\rho)) = p_0 \cdot p_1 \dots p_{k-1}$ , where  $p_\ell = p_{e_\ell}$ , for  $0 \leq \ell < k$ . This is inspired by the Markov chain case in which the probability of a basic cylinder is defined to be the product of the probabilities of the events encountered along the common finite prefix of the basic cylinder. However, showing directly that this extends canonically to a probability measure over  $\widehat{SA}_{\mathbf{s}}$  is very difficult.

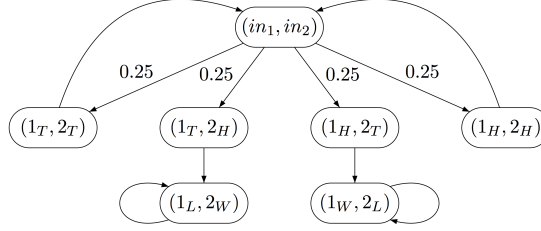
We shall tackle this problem by associating a Markov chain  $\mathcal{M}$  with  $\mathcal{D}$  and then embedding  $\widehat{SA}_{\mathbf{s}}$  into  $SA_{\mathbf{s}}$ , the  $\sigma$ -algebra generated by the infinite paths in  $\mathcal{M}$  starting from  $\mathbf{s}$ . The standard probability measure over  $SA_{\mathbf{s}}$  will then induce a probability measure over  $\widehat{SA}_{\mathbf{s}}$ .

## 4 The Markov chain semantics

We associate a Markov chain with a DMC using the notion of maximal steps with respect to the trace alphabet  $(\Sigma, I)$ . A *nonempty* set of events  $u \subseteq \Sigma$  is

<sup>4</sup> For infinite sequences, it is technically more convenient to define traces using projection equivalence rather than permutation of independent actions.

$\mathbf{s}'$



**Fig. 2.** Markov chain for the DMC in Fig 1

a *step* at  $\mathbf{s}$  iff each  $e \in u$  is enabled at  $\mathbf{s}$  and for every distinct pair of events  $e, e' \in u$ ,  $e \perp e'$ . We say  $u$  is a *maximal* step at  $\mathbf{s}$  iff  $u$  is a step at  $\mathbf{s}$  and  $u \cup \{e\}$  is not a step at  $\mathbf{s}$  for any  $e \notin u$ . In Fig. 1,  $\{e_h, e'_h\}$ ,  $\{e_h, e'_t\}$ ,  $\{e_t, e'_h\}$  and  $\{e_t, e'_t\}$  are maximal steps at the initial state  $(in_1, in_2)$ .

Let  $u$  be a maximal step at  $\mathbf{s}$ . Then  $\mathbf{s}'$  is the  $u$ -successor of  $\mathbf{s}$  iff the following conditions are satisfied: **(i)** For each  $e \in u$ , if  $e = (\mathbf{v}, a, \mathbf{v}')$  and  $i \in loc(e)$  then  $\mathbf{s}'_i = \mathbf{v}'_i$ , and **(ii)**  $\mathbf{s}'_j = \mathbf{s}_j$  if  $j \notin loc(u)$ , where, as usual,  $loc(u) = \bigcup_{e \in u} loc(e)$ .

Suppose  $u$  is a maximal step at  $\mathbf{s}$  and  $i \in loc(u)$ . Then, because events in a step are independent, it follows that there exists a unique  $e \in u$  such that  $i \in loc(e)$ , so the  $u$ -successor of  $\mathbf{s}$  is unique. We say  $\mathbf{s}'$  is a *successor* of  $\mathbf{s}$  iff there exists a maximal step  $u$  at  $\mathbf{s}$  such that  $\mathbf{s}'$  is the  $u$ -successor of  $\mathbf{s}$ . From the definition of a DMC, it is easy to see that if  $\mathbf{s}'$  is a successor of  $\mathbf{s}$  then there exists a unique maximal step  $u$  at  $\mathbf{s}$  such that  $\mathbf{s}'$  is the  $u$ -successor of  $\mathbf{s}$ . Finally, we say that  $\mathbf{s}$  is a *deadlock* iff no event is enabled at  $\mathbf{s}$ .

**Definition 4.** The Markov chain  $\mathcal{M} : \mathbf{S} \times \mathbf{S} \rightarrow [0, 1]$  generated by  $\mathcal{D}$  is given by:

- If  $\mathbf{s} \in \mathbf{S}$  is a deadlock then  $\mathcal{M}(\mathbf{s}, \mathbf{s}) = 1$  and  $\mathcal{M}(\mathbf{s}, \mathbf{s}') = 0$  if  $\mathbf{s} \neq \mathbf{s}'$ .
- Suppose  $\mathbf{s} \in \mathbf{S}$  is not a deadlock. Then  $\mathcal{M}(\mathbf{s}, \mathbf{s}') = p$  iff there exists a maximal step  $u$  at  $\mathbf{s}$  such that  $\mathbf{s}'$  is the  $u$ -successor of  $\mathbf{s}$  and  $p = \prod_{e \in u} p_e$ .
- If  $\mathbf{s}$  is not a deadlock and  $\mathbf{s}'$  is not a successor of  $\mathbf{s}$  then  $\mathcal{M}(\mathbf{s}, \mathbf{s}') = 0$ .

It follows that  $\mathcal{M}(\mathbf{s}, \mathbf{s}') \in [0, 1]$  for every  $\mathbf{s}, \mathbf{s}' \in \mathbf{S}$ . In addition, if  $u$  and  $u'$  are two maximal steps at  $\mathbf{s}$  then  $loc(u) = loc(u')$  and  $|u| = |u'|$ . Using these facts it is easy to verify that  $\mathcal{M}$  is indeed a finite state Markov chain. The initial state of  $\mathcal{M}$  is  $\mathbf{s}^{in} = (s_1^{in}, s_2^{in}, \dots, s_n^{in})$ .

In Fig. 2 we show the Markov chain of the DMC whose transition system was shown in Fig. 1. Again, unlabelled transitions have probability 1. A DMC may have a *reachable* deadlock—a sequence of global states  $\mathbf{s}_0 \mathbf{s}_1 \dots \mathbf{s}_k$  such that  $\mathbf{s}_0 = \mathbf{s}^{in}$ ,  $\mathcal{M}(\mathbf{s}_\ell, \mathbf{s}_{\ell+1}) > 0$ , for  $0 \leq \ell < k$ , and  $\mathbf{s}_k$  is a deadlock. One can effectively check for reachable deadlocks by forming a *non-deterministic* asynchronous transition system and analyze it using traditional verification methods. Henceforth, for simplicity, we assume that DMCs are free of (reachable) deadlocks. Our results can be easily extended to handle deadlocks.

Suppose  $u$  is a maximal step at  $\mathbf{s}$  with  $|u| = m$  and  $|S_i| = k$  for each  $i \in loc(u)$ . In  $\mathcal{M}$  there will be, in general,  $k^m$  transitions at  $\mathbf{s}$ . In contrast there will be at most  $k \cdot m$  transitions at  $\mathbf{s}$  in  $TS$ . Hence—assuming that we do not explicitly

$\mathbf{s}'$

construct  $\mathbf{S}$ —there can be substantial computational gains if one can verify the properties of  $\mathcal{D}$  by working with  $TS$  instead of  $\mathcal{M}$ . This will become clearer when we look at some larger examples in Section 8.

**The path space of  $\mathcal{M}$**  Let  $\mathcal{M}$  be the Markov chain associated with a DMC  $\mathcal{D}$ . A finite path in  $\mathcal{M}$  from  $\mathbf{s}$  is a sequence  $\tau = \mathbf{s}_0 \mathbf{s}_1 \dots \mathbf{s}_m$  such that  $\mathbf{s}_0 = \mathbf{s}$  and  $\mathcal{M}(\mathbf{s}_\ell, \mathbf{s}_{\ell+1}) > 0$ , for  $0 \leq \ell < m$ . The notion of an infinite path starting from  $\mathbf{s}$  is defined as usual.  $Path_{\mathbf{s}}$  and  $Path_{\mathbf{s}}^{fin}$  denote the set of infinite and finite paths starting from  $\mathbf{s}$ , respectively.

For  $\tau \in Path_{\mathbf{s}}^{fin}$ ,  $\uparrow\tau \subseteq Path_{\mathbf{s}}$  is the set of infinite paths that have  $\tau$  as a prefix.  $\Upsilon \subseteq Path_{\mathbf{s}}$  is a basic cylinder at  $\mathbf{s}$  if  $\Upsilon = \uparrow\tau$  for some  $\tau \in Path_{\mathbf{s}}^{fin}$ . The  $\sigma$ -algebra over  $Path_{\mathbf{s}}$ , denoted  $SA(\mathbf{s})$ , is the least family that contains the basic cylinders at  $\mathbf{s}$  and is closed under countable unions and complementation (relative to  $Path_{\mathbf{s}}$ ).  $P_{\mathbf{s}} : SA(\mathbf{s}) \rightarrow [0, 1]$  is the usual probability measure that assigns to each basic cylinder  $\uparrow\tau$ , with  $\tau = \mathbf{s}_0 \mathbf{s}_1 \dots \mathbf{s}_m$ , the probability  $p = p_0 \cdot p_1 \dots p_{m-1}$ , where  $\mathcal{M}(\mathbf{s}_\ell, \mathbf{s}_{\ell+1}) = p_\ell$ , for  $0 \leq \ell < m$ .

## 5 The probability measure for the trajectory space

To construct a probability measure over the trajectory space we shall associate infinite paths in  $\mathcal{M}$  with maximal trajectories in  $TS$ . The Foata normal form from Mazurkiewicz trace theory will help achieve this. Let  $\xi \in \Sigma^*$ . A standard fact is that  $[\xi]$  can be canonically represented as a “step” sequence of the form  $u_1 u_2 \dots u_k$ . More precisely, the Foata normal form of the finite trace  $[\xi]$ , denoted  $FN([\xi])$ , is defined as follows [8].

- $FN([\epsilon]) = \epsilon$ .
- Suppose  $\xi = \xi' e$  and  $FN([\xi']) = u_1 u_2 \dots u_k$ . If there exists  $e' \in u_k$  such that  $(e', e) \notin I$  then  $FN([\rho]) = u_1 u_2 \dots u_k \{e\}$ . If not, let  $\ell$  be the least integer in  $\{1, 2, \dots, k\}$  such that  $e I e'$  for every  $e' \in \bigcup_{\ell \leq m \leq k} u_m$ . Then  $FN([\rho]) = u_1 \dots u_{\ell-1} (u_\ell \cup \{e\}) u_{\ell+1} \dots u_m$ .

For the example shown in Fig. 1,  $FN(e_h e'_t h t \ell' w w) = \{e_h, e'_t\} \{ht\} \{w, \ell'\} \{w\}$ . This notion is extended to infinite traces in the obvious way. Note that  $\xi \approx \xi'$  iff  $FN(\xi) = FN(\xi')$ .

Conversely, we can extract a (maximal) step sequence from a path in  $\mathcal{M}$ . Suppose  $\mathbf{s}_0 \mathbf{s}_1 \dots$  is a path in  $Path_{\mathbf{s}}$ . There exists a unique sequence  $u_1 u_2 \dots$  such that  $u_\ell$  is a maximal step at  $\mathbf{s}_{\ell-1}$  and  $\mathbf{s}_\ell$  is the  $u_\ell$ -successor of  $\mathbf{s}_{\ell-1}$  for every  $\ell > 0$ . We let  $st(\tau) = u_1 u_2 \dots$  and call it the step sequence induced by  $\tau$ .

This leads to the map  $tp : Trj_{\mathbf{s}} \rightarrow Path_{\mathbf{s}}$  given by  $tp(\rho) = \tau$  iff  $FN(ev(\rho)) = st(\tau)$ . It is easy to check that  $tp$  is well-defined. As usual, for  $X \subseteq Trj_{\mathbf{s}}$  we define  $tp(X) = \{tp(\rho) \mid \rho \in X\}$ . It turns out that  $tp$  maps each basic cylinder in the trajectory space to a finite union of basic cylinders in the path space. As a result,  $tp$  maps every measurable set of trajectories to a measurable set of paths. Consequently, one can define the probability of a measurable set of trajectories  $X$  to be the probability of the measurable set of paths  $tp(X)$ .



$\mathbf{s}$ ,

To understand how  $tp$  acts on the basic cylinder  $BC(\rho)$ , let  $FN(ev(\rho)) = u_1u_2 \dots u_k$ . We associate with  $\rho$  the set of finite paths  $paths(\rho) = \{\pi \mid st(\pi) = U_1U_2 \dots U_k \text{ and } u_\ell \subseteq U_\ell \text{ for } 1 \leq \ell \leq k\}$ . In other words  $\pi \in paths(\rho)$  if it extends each step in  $FN(ev(\rho))$  to a maximal step. Then,  $tp$  maps  $BC(\rho)$  to the (finite) union of the basic cylinders in  $paths(\rho)$ . These observations and their main consequence, namely the construction of a probability measure over the trajectory space, can be summarized as:

**Lemma 5.**

- (i) Let  $B = BC(\rho)$  be a basic *trj*-cylinder from  $\mathbf{s}$ , with  $FN(ev(\rho)) = u_1u_2 \dots u_k$ . Then  $tp(B)$  is a finite union of basic cylinder sets in  $SA(\mathbf{s})$  and is hence a member of  $SA(\mathbf{s})$ . Furthermore  $P(tp(B)) = \prod_{1 \leq \ell \leq k} p_\ell$  where  $p_\ell = \prod_{e \in u_\ell} p_e$  for  $1 \leq \ell \leq k$ .
- (ii) If  $B \in \widehat{SA}(\mathbf{s})$  then  $tp(B) \in SA(\mathbf{s})$ .
- (iii) Define  $\widehat{P} : \widehat{SA}(\mathbf{s}) \rightarrow [0, 1]$  as  $\widehat{P}(B) = P(tp(B))$ . Then  $\widehat{P}$  is a probability measure over  $\widehat{SA}(\mathbf{s})$ .

*Proof sketch* Let  $BC(\rho)$  be the basic *trj*-cylinder from  $\mathbf{s}$  generated by  $\rho \neq \epsilon$  and  $FN(ev(\rho)) = u_1u_2 \dots u_k$ . Suppose  $\tau \in Path_{\mathbf{s}}$ . Then, using the semantic definitions, it is tedious but straightforward to show that  $\tau \in tp(BC(\rho))$  iff  $u_i \subseteq st(\tau)(\ell)$ , for  $1 \leq \ell \leq k$ . (Here,  $st(\tau)(\ell)$  is the maximal step appearing in position  $\ell$  of the sequence  $st(\tau)$ .) It will then follow that  $tp(BC(\rho))$  is a finite union of basic cylinder sets in  $SA(\mathbf{s})$  and is hence a member of  $SA(\mathbf{s})$ . Furthermore, one can argue that  $P(tp(BC(\rho))) = \prod_{1 \leq \ell \leq k} p_\ell$ .

For the other two parts, we first establish easily that if  $B \in \widehat{SA}(\mathbf{s})$ ,  $\rho \in B$  and  $\rho \simeq \rho'$  then  $\rho' \in B$  as well. Next, it is straightforward to show that if  $B, B' \in \widehat{SA}(\mathbf{s})$  with  $B \cap B' = \emptyset$  then  $tp(B) \cap tp(B') = \emptyset$  too. Finally, one can also show  $tp$  is onto. Using these facts, the second and third parts of the lemma can be easily established.  $\square$

Note that while a finite path in  $\mathcal{M}$  always induces a maximal step sequence, a finite trajectory, in general, does not have this structure. Some components can get ahead of others by an arbitrary amount. The lemma above states that, despite this, any finite trajectory defines a basic cylinder whose probability can be easily computed. This helps considerably when verifying the properties of  $\mathcal{M}$ . In particular local reachability properties can be checked by exercising only those components that are relevant.

Going back to our running example let  $\rho_t = (in_1, in_2)e_t(T_1, in_2)$ , and  $X_t = \uparrow\rho_t$ . Let  $\rho'_t = (in_1, in_2)e'_t(in_1, T_2)$ , and  $X'_t = \uparrow\rho'_t$ . Assume  $\rho_h, X_h, \rho'_h$  and  $X'_h$  are defined similarly. Then  $\widehat{P}(X_t) = \widehat{P}(X'_h) = 0.5$  while  $\widehat{P}(X_h \cup X_t) = 1$ . On the other hand due to the fact that  $e_h$  and  $e'_h$  are independent we will have  $\widehat{P}(X_h \cup X'_h) = 0.75$ .

## 6 Model checking $PBLTL^\otimes$ specifications

We have designed a statistical model checking procedure to verify dynamic properties of DMCs. The specification logic  $PBLTL^\otimes$  (product  $PBLTL$ ) is a simple

$\mathbf{s}'$

generalization of probabilistic bounded linear time temporal logic (*PBLTL*) [17] that captures boolean combinations of local properties of the components. The logic can express interesting global reachability properties as well as the manner in which the components influence each other.

We assume a collection of pairwise disjoint sets of atomic propositions  $\{AP_i\}$ . As a first step, the formulas of  $BLTL^\otimes$  are given as follows.

- (i)  $ap \in AP_i$  is a  $BLTL^\otimes$  formula and  $type(ap) = \{i\}$ .
- (ii) If  $\varphi$  and  $\varphi'$  are  $BLTL^\otimes$  formulas with  $type(\varphi) = type(\varphi') = \{i\}$  then so is  $\varphi \mathbf{U}_i^t \varphi'$  where  $t$  is a non-negative integer. Further,  $type(\varphi \mathbf{U}_i^t \varphi') = \{i\}$ . As usual,  $\Diamond^t \varphi$  abbreviates  $true \mathbf{U}_i^t \varphi$  and  $\Box^t \varphi$  is defined as  $\neg \Diamond^t \neg \varphi$ .
- (iii) If  $\varphi$  and  $\varphi'$  are  $BLTL^\otimes$  formulas then so are  $\neg \varphi$  and  $\varphi \vee \varphi'$  with  $type(\neg \varphi) = type(\varphi)$  and  $type(\varphi \vee \varphi') = type(\varphi) \cup type(\varphi')$ .

The formulas of  $PBLTL^\otimes$  are given by:

- (i) Suppose  $\varphi$  is a  $BLTL^\otimes$  formula and  $\gamma$  a rational number in the open interval  $(0, 1)$ . Then  $Pr_{\geq \gamma}(\varphi)$  is a  $PBLTL^\otimes$  formula.
- (ii) If  $\psi$  and  $\psi'$  are  $PBLTL^\otimes$  formulas then so are  $\neg \psi$  and  $\psi \vee \psi'$ .

To define the semantics, we project each trajectory to its components. For  $\mathbf{s} \in \mathbf{S}$  and  $i \in [n]$  we define  $Proj_i : Trj_{\mathbf{S}}^{fin} \rightarrow S_i^+$  inductively.

- (i)  $Proj_i(\mathbf{s}) = \mathbf{s}_i$ .
- (ii) Suppose  $\rho = \mathbf{s}_0 e_0 \mathbf{s}_1 \dots \mathbf{s}_m e_m \mathbf{s}_{m+1}$  is in  $Trj_{\mathbf{S}}^{fin}$  and  $\rho' = \mathbf{s}_0 e_0 \mathbf{s}_1 \dots \mathbf{s}_m$ . If  $i \in loc(e_m)$  then  $Proj_i(\rho) = Proj_i(\rho')(\mathbf{s}_{m+1})_i$ . Otherwise  $Proj_i(\rho) = Proj_i(\rho')$ .

We lift  $Proj_i$  to infinite trajectories in the obvious way—note that  $Proj_i(\rho)$  can be a finite sequence for the infinite trajectory  $\rho$ . We assume a set of local valuation functions  $\{V_i\}$ , where  $V_i : S_i \rightarrow 2^{AP_i}$ . Let  $\varphi$  be a  $BLTL^\otimes$  formula with  $type(\varphi) = \{i\}$ . We begin by interpreting such formulas over sequences generated by the alphabet  $S_i$ . For  $\varrho \in S_i^+ \cup S_i^\omega$ , the satisfaction relation  $\varrho, k \models_i \varphi$ , with  $0 \leq k \leq |\varrho|$ , is defined as follows.

- (i)  $\varrho, k \models_i ap$  for  $ap \in AP_i$  iff  $ap \in V_i(\varrho(k)(i))$ , where  $\varrho(k)(i)$  is the  $S_i$ -state at position  $k$  of the sequence  $\varrho$ .
- (ii)  $\neg$  and  $\vee$  are interpreted in the usual way.
- (iii)  $\varrho, k \models_i \varphi_1 \mathbf{U}_i^t \varphi_2$  iff there exists  $\ell$  such that  $k \leq \ell \leq \max(k + t, |\varrho|)$  with  $\varrho, \ell \models_i \varphi_2$ , and  $\varrho, m \models_i \varphi_1$ , for  $k \leq m < \ell$ .

As usual,  $\varrho \models_i \varphi$  iff  $\varrho, 0 \models_i \varphi$ . Next, suppose  $\varphi$  is a  $BLTL^\otimes$  formula and  $\rho \in Path_{\mathbf{S}}$ . Then the relation  $\rho \models_{\mathbf{S}} \varphi$  is defined as follows.

- (i) If  $type(\varphi) = \{i\}$  then  $\rho \models_{\mathbf{S}} \varphi$  iff  $Proj_i(\rho) \models_i \varphi$ .
- (ii) Again,  $\neg$  and  $\vee$  are interpreted in the standard way.

Given a formula  $\varphi$  in  $BLTL^\otimes$  and a global state  $\mathbf{s}$ , we define  $Trj_{\mathbf{S}}(\varphi)$  to be the set of trajectories  $\{\rho \in Trj_{\mathbf{S}} \mid \rho \models_{\mathbf{S}} \varphi\}$ .

**Lemma 6.** *For every formula  $\varphi$ ,  $Trj_{\mathbf{S}}(\varphi)$  is a member of  $\widehat{SA}(\mathbf{s})$ .*

$\mathbf{s}'$

*Proof sketch* If we interpret the formulas over  $\mathcal{M}$ , we easily derive that  $Path_{\mathbf{s}}(\varphi)$  is a member of  $SA(\mathbf{s})$  for every  $\varphi$ . We then use Lemma 5 to obtain this result.  $\square$

The semantics of  $PBLTL^{\otimes}$  is now given by the relation  $\mathcal{D} \models_{\mathbf{s}}^{trj} \psi$ , defined as:

- (i) Suppose  $\psi = Pr_{\geq \gamma}(\varphi)$ . Then  $\mathcal{D} \models_{\mathbf{s}}^{trj} \psi$  iff  $\hat{P}(Path_{\mathbf{s}}(\varphi)) \geq \gamma$ .
- (ii) Again, the interpretations of  $\neg$  and  $\vee$  are the standard ones.

For the example in Fig. 1, one can assert  $\hat{P}_{\geq 0.99}((\diamond^7 L_1 \wedge \diamond^7 W_2) \vee (\diamond^7 W_1 \wedge \diamond^7 L_2))$ . Here the local states also serve as the atomic propositions. Hence, the formula says that with probability  $\geq 0.99$ , a winner will be decided within 7 rounds.

We write  $\mathcal{D} \models^{trj} \psi$  for  $\mathcal{D} \models_{\mathbf{s}^{in}}^{trj} \psi$ . The model checking problem is to determine whether  $\mathcal{D} \models^{trj} \psi$ . We shall adapt the SMC procedure developed in [27] to solve this problem approximately.

## 7 Statistical model checking

Given a DMC  $\mathcal{D}$  and a  $PBLTL^{\otimes}$  specification  $\psi$ , our goal is to determine whether  $\mathcal{D} \models^{trj} \psi$  (that is,  $\mathcal{D} \models_{\mathbf{s}^{in}}^{trj} \psi$ ). We develop a statistical model checking (SMC) procedure to provide an approximate solution to this problem. We note that in the Markov chain setting, given a BLTL formula and a path in the chain, there is a bound  $k$  that depends only on the formula such that we can decide whether the path is a model of the formula by examining just a prefix of length  $k$  of the path [17]. By the same reasoning, for a  $BLTL^{\otimes}$  formula  $\varphi$ , we can compute a vector of bounds  $(k_1, k_2, \dots, k_n)$  that depends only on  $\varphi$  such that for any trajectory  $\rho$  starting from  $\mathbf{s}^{in}$ , we only need to examine a finite prefix  $\rho'$  of  $\rho$  that satisfies  $|Proj_i(\rho')| \geq k_i$ , for  $1 \leq i \leq n$ . The complication in our setting is that such a prefix of  $\rho$  may not exist.

To cope with this, we maintain a count vector  $(c_1, c_2, \dots, c_n)$  that records how many times each component has moved along the trajectory  $\rho$  that has been generated so far. A simple reachability analysis will reveal whether a component is *dead* in the current global state; that is, starting from the current state, there is no possibility of reaching a state in which an event involving this agent can be executed. We mark such components as dead. We then execute, one by one, all the enabled actions—using a fixed linear order over the set of actions—followed by one move by each of the participating agents according to the underlying probabilities. Recall that action  $a$  is enabled at  $\mathbf{s}$  iff  $\mathbf{s}_a \in en_a$ . Due to the determinacy of communication, the global state thus reached will depend only on the probabilistic moves chosen by the participating agents. We then update the count vector to  $(c'_1, c'_2, \dots, c'_n)$  and mark the new dead components. It is not difficult to prove that, continuing in this manner, with probability 1 we will eventually generate a finite trajectory  $\hat{\rho}$  and reach a global state  $\mathbf{s}$  with count vector  $(\hat{c}_1, \hat{c}_2, \dots, \hat{c}_n)$  such that for each component  $i$ , either  $\hat{c}_i \geq k_i$  or  $i$  is dead at  $\mathbf{s}$ . We then check if  $\hat{\rho}$  satisfies  $\varphi$  and update the score associated with the statistical test described below.

The parameters for the test are  $\delta, \alpha, \beta$ , where  $\delta$  is the size of the indifference region and  $(\alpha, \beta)$  is the strength of the test, with  $\alpha$  bounding the Type I errors (false positives) and  $\beta$  bounding the Type II errors (false negatives). These parameters are to be chosen by the user. We generate finite i.i.d. sample trajectories sequentially. We associate a Bernoulli random variable  $x_\ell$  with the sample  $\rho_\ell$  and set  $x_\ell = 1$  if  $\rho_\ell \in Trj_{s^{in}}(\varphi)$  and set  $x_\ell = 0$  otherwise. We let  $c_m = \sum_\ell x_\ell$  and compute the score  $SPRT$  via

$$SPRT = \frac{(\gamma^+)^{c_m} (1 - \gamma^+)^{n - c_m}}{(\gamma^-)^{c_m} (1 - \gamma^-)^{n - c_m}}$$

Here  $\gamma^+ = \gamma + \delta$  and  $\gamma^- = \gamma - \delta$ . If  $SPRT \geq \frac{1 - \beta}{\alpha}$ , we declare  $\mathcal{D} \models^{trj} \hat{P}_{\geq r} \varphi$ . If  $SPRT \leq \frac{\beta}{1 - \alpha}$ , we declare  $\mathcal{D} \not\models^{trj} \hat{P}_{\geq r} \varphi$ . Otherwise, we draw one more sample and repeat.

This test is then extended to handle formulas of the form  $\neg\psi$  and  $\psi_1 \vee \psi_2$  in the usual way [17]. It is easy to establish the correctness of this statistical model checking procedure.

## 8 Experimental results

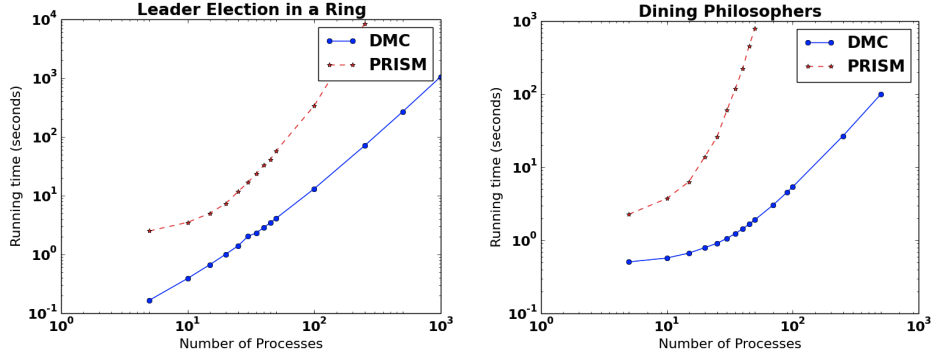
We have tested our statistical verification procedure on two probabilistic distributed algorithms: (i) a leader election protocol for a unidirectional ring of anonymous processes by Itai and Rodeh [10, 14] and (ii) a randomized solution to the dining philosophers problem by Lehman and Rabin [20].

For comparison with Markov chain based verification we used the probabilistic model checking tool PRISM, which tackles these two examples as case studies [13, 24]. Since PRISM does not currently support SMC for BLTL specifications, we used the simulation based approximate verification feature of PRISM. We compared the time taken by our SMC procedure with that of approximate verification in PRISM for roughly the same number of simulations.

In the leader election protocol, each process randomly chooses an identity from  $\{1, 2, \dots, N\}$ , and passes it on to its neighbour. If a process receives an identity lower than its own, the message is dropped. If the identity is higher than its own, the process drops out of the election and forwards the message. Finally, if the identity is the same as its own, the process forwards the message, noting the identity clash. If an identity clash is recorded, all processes with the highest identity choose a fresh identity and start another round.

We have built a DMC model of this system in which each process and channel is an agent. Messages are transferred between processes and channels via synchronizations. For simplicity, all channels in our implementation have capacity 1. We can easily construct higher capacity channels by cascading channels of capacity 1 while staying within the DMC formalism.

The challenge in modelling the dining philosophers problem as a DMC is to represent the forks between philosophers, which are typically modelled as shared variables. We use a deterministic round robin protocol to simulate shared variables. The same technique can be used for a variety of other randomized distributed algorithms presented as case studies for PRISM.



**Fig. 3.** Comparison of simulation times in DMC and PRISM

We ran our trajectories based SPRT procedure on a Linux server (Intel Xeon 2.30 GHz, 16 core, 72GB RAM). For the first example, we verified that a leader is elected with probability above 0.99 within  $N$  rounds, for a ring of  $N$  processes, upto  $N = 1000$ . For the dining philosophers, we verified that every philosopher eventually eats, upto  $N = 500$  philosophers. This property could not be checked using approximate verification in PRISM because it exceeded the simulation bounds even for  $N = 11$ . To compare with PRISM, we introduced a boolean variable that is set when a philosopher eats for the first time and verified the property that with probability above 0.95, a fixed fraction (0.4) of the philosophers have eaten at least once within a bounded number of steps.

We tested the same properties on the PRISM model, using simulation based approximate verification. For a fair comparison, we ensured that the number of simulation runs in PRISM were approximately the same as the average number of simulation runs required for SPRT verification in the DMC model.

In Fig. 3, we have compared the running time for SPRT model-checking in the DMC model against the running time for approximate verification in PRISM. The  $x$ -axis is the number of processes in the system and the  $y$ -axis is the running time, in seconds. Both axes are rendered in a log scale. In PRISM, we could not check the leader election property beyond  $N = 250$  and, for the dining philosophers example, the largest system we could run had  $N = 50$ . The experiments show that simulations using asynchronous trajectories are 10 to 100 times faster.

## 9 Conclusion

We have formulated a distributed probabilistic system model called DMCs. Our model achieves a clean mix of concurrency and probabilistic dynamics by restricting synchronizations to be deterministic. Our key technical contribution is the construction of a probability measure over the  $\sigma$ -algebra generated by the (interleaved) trajectories of a DMC. This opens up the possibility of using partial order reduction techniques to efficiently verify the dynamic properties of a

DMC. As a first step in this direction we have developed a SPRT based statistical model checking procedure for the logic  $PBLTL^\otimes$ . Our experiments suggest that our method can handle systems of significant sizes.

The main partial order concept we have used is to group trajectories into equivalence classes. One can also explore how ample sets [6] and related notions can be used to model check properties specified in logics such as PCTL [3]. Another possibility is to see if the notion of finite unfoldings from Petri net theory can be applied in the setting of DMCs [9, 21].

In the two examples we have discussed, the specification has a global character, since it mentions every agent in the system. In many specifications, only a few agents may be mentioned. If the system is loosely coupled, we can check whether the required property is fulfilled without having to exercise all the agents. This will lead to additional computational gains.

One observation is that in standard randomized distributed algorithms, like the case studies in [24], probabilistic moves are always local. The DMC model allows synchronous probabilistic moves where the probability distribution is influenced by information obtained through communication. This allows us to model situations such as an actuator for an implanted device making probabilistic transitions based on information received from external sensors. In such situations, the overall dynamics would typically be too complex to explore exhaustively using traditional model checking techniques, but statistical model checking can be applied provided we can perform efficient simulations, which we have demonstrated is possible with DMCs.

We currently allow agents to gain complete information about the state of the agents they synchronize with. In practice, only a part of this state may/should be exposed. An interesting extension would be point-to-point asynchronous communications through bounded buffers using a finite message alphabet.

Finally, though almost all the case studies in [24] can be recast as DMCs, it will be fruitful to understand the theoretical and practical limitations of deterministic communications in a distributed probabilistic setting.

## References

1. S. Abbes and A. Benveniste. True-concurrency probabilistic models: Branching cells and distributed probabilities for event structures. *Inf. Comput.*, 204(2):231–274, 2006.
2. S. Abbes and A. Benveniste. True-concurrency probabilistic models: Markov nets and a law of large numbers. *Theor. Comput. Sci.*, 390(2-3):129–170, 2008.
3. C. Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
4. J. Bogdoll, L.M.F. Fioriti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FMOODS/FORTE*, pages 59–74, 2011.
5. E.M. Clarke, J.R. Faeder, C.J. Langmead, L.A. Harris, S.K. Jha, and A. Legay. Statistical model checking in BioLab: Applications to the automated analysis of t-cell receptor signaling pathway. In *CMSB*, pages 231–250, 2008.
6. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.

7. A. David, K.G. Larsen, A. Legay, M. Mikučionis, and Z. Wang. Time for statistical model checking of real-time systems. In *Computer Aided Verification*, pages 349–355. Springer, 2011.
8. V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific, 1995.
9. J. Esparza and K. Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking*. Monographs in theoretical computer science. Springer-Verlag Berlin Heidelberg, 2008.
10. W. Fokkink and J. Pang. Variations on Itai-Rodeh leader election for anonymous rings and their analysis in PRISM. *J. of Universal Computer Science*, 12(8):981–1006, 2006.
11. M. Größer and C. Baier. Partial order reduction for Markov Decision Processes: A survey. In *FMCO*, pages 408–427, 2005.
12. R. Grosu, X. Huang, S.A. Smolka, W. Tan, and S. Tripakis. Deep random search for efficient model checking of timed automata. In Fabrice Kordon and Oleg Sokolsky, editors, *Monterey Workshop*, volume 4888 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2006.
13. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *12th TACAS*, pages 441–444, 2006.
14. A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1):60–87, 1990.
15. C. Jegourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking–PLASMA. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 498–503, 2012.
16. S. Jha. Statistical analysis of privacy and anonymity guarantees in randomized security protocol implementations. *arXiv preprint arXiv:0906.5110*, 2009.
17. S.K. Jha, E.M. Clarke, C.J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian approach to model checking biological systems. In *CMSB*, pages 218–234, 2009.
18. C.J. Langmead and S.K. Jha. Predicting protein folding kinetics via temporal logic model checking. In *WABI*, pages 252–264, 2007.
19. A. Legay and M. Viswanathan. Simulation + hypothesis testing for model checking of probabilistic systems. In *QEST*, page 3. IEEE Computer Society, 2009.
20. D. Lehmann and M. Rabin. On the advantage of free choice: A symmetric and fully distributed solution to the dining philosophers problem (extended abstract). In *Proc. 8th Annual ACM Symposium on Principles of Programming Languages (POPL’81)*, pages 133–138, 1981.
21. K.L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
22. G. Pighizzini S. Jesi and N. Sabadini. Probabilistic asynchronous automata. *Mathematical Systems Theory*, 29(1):5–31, 1996.
23. D. Varacca, H. Völzer, and G. Winskel. Probabilistic event structures and domains. *Theor. Comput. Sci.*, 358(2-3):173–199, 2006.
24. Various. PRISM case studies. <http://www.prismmodelchecker.org/casestudies>, 2013.
25. A. Wald. *Sequential Analysis*. John Wiley and Sons, 1st edition, 1947.
26. H.L.S. Younes. *Verification and planning for stochastic processes with asynchronous events*. PhD thesis, Pittsburgh, PA, USA, 2004. AAI3159989.
27. H.L.S. Younes and R.G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, pages 223–235, 2002.
28. W. Zielonka. Notes on finite asynchronous automata. *ITA*, 21(2):99–135, 1987.